Interactive pedagogical programs based on constraint grammar

Lene Antonsen
University of Tromsø
Norway
lene.antonsen
@uit.no

Saara Huhmarniemi
University of Tromsø
Norway
saara.huhmarniemi
@helsinki.fi

Trond Trosterud
University of Tromsø
Norway
trond.trosterud
@uit.no

Abstract

This article presents a set of interactive parser-based CALL programs for North Sámi. The programs are based on a finite state morphological analyser and a constraint grammar parser which is used for syntactic analysis and navigating in the dialogues. The analysers provide effective and reliable handling of a wide variety of user input. In addition, relaxation of the grammatical analysis of the user input enables locating grammatical errors and reacting to the errors with appropriate feedback messages.

1 Introduction

This paper describes the implementation a set of CALL (Computer Assisted Language Learning) programs for learners of North Sámi (a Uralic language), based on a finite state transducer (fst) and constraint grammar (CG) technology.

The pedagogical programs are available on a web-based learning platform OAHPA!, accessible at `http:\\oahpa.uit.no`. There are six programs altogether: A word quiz (Leksa), a numeral quiz (Numra), basic morphological exercises (Morfa-S), morphological exercises in a sentential frame (Morfa-C), a question-answer (QA) drill (Vasta), and a dialogue program (Sahka).

The OAHPA! platform is implemented in Django, a Python-based web development framework, combined with a Mysql database.

In section 2 we describe the initial linguistic resources and the pedagogical motivation behind the programs. Section 3 presents the pedagogical lexicon and the morphological analyser. The fourth section presents the parser-based CALL programs and shows how the CG-parser was utilised for error detection and navigation in the programs accepting free sentence input. Section 5 contains an evaluation of the programs.

2 Background

2.1 Basic grammatical analysis

The pedagogical programs in OAHPA! are based upon three pre-existing language technology resources developed at the University of Tromsø: a morphological analyser/generator, a CG parser for North Sámi and a number word generator compiled with the Xerox compiler xfst.

The morphological analyser/generator is implemented with fst and compiled with the Xerox compilers twolc and lexc (Beesley and Karttunen, 2003). Sámi languages have large morphological paradigms for each lexeme – verbs and adjectives have more than 100 inflected forms. In addition, some of the paradigm members have a very low text frequency. Due to the limited amount of electronically available text resources, an fst analyser was used, rather than e.g. an HMM tagger (Trosterud, 2007). The lexicon contains 97.500 lemmata – almost half of them proper nouns. We made two different variants of the analyser/generator: one tolerant, with morphological patterns based upon actual usage, and the other one normative, adhering to the written standard.

The morphological disambiguator is implemented in the CG-framework (Karlsson et. al, 1995). The CG-framework is based upon manually written rule sets and a syntactic analyser which selects the correct analysis in case of homonymy and adds grammatical function and dependency relations to the analysis. We used vislcg3 for the compilation of CG rules. Vislcg3 is a new, improved version of the open source compiler vislcg (visl, 2008). The CG-framework is presented in section 4.1.

2.2 Previous accounts on parser-based CALL

Even if many interactive parser-based CALL programs are described in the literature, see (Gamper and Knapp, 2002; Heift and Schulze, 2007), very few of them are available for actual use online and most systems have remained at a prototype level. One of very few exceptions is e-tutor, a program for teaching German to foreigners (Heift, 2001; Heift and Nicholson, 2001), at `http://e-tutor.org`. e-tutor gives very good feedback to student's errors, but the possible input is restricted to small, fixed vocabularies, and there is no dialogue. The grammar formalism used is Head-driven Phrase Structure Grammar (HPSG).

Vislcg3 is used in the VISL-suite of games for teaching grammatical analysis on the Internet `http://visl.sdu.dk`. Most of the games in VISL are based on pre-analysed sentences, but one of the programs accepts free user input in some of the 7 supported languages. The input is analysed or changed into grammar exercises (Bick, 2005).

2.3 The pedagogical motivation

The main goal of the development of OAHPA! was to develop a language tutoring system going beyond simple multiple-choice questions or string matching algorithms, with free-form dialogues and sophisticated error analysis. Immediate error feedback and advice about morphology and grammar were seen as important requirements for the program.

In addition, the programs were designed to be flexible so that the student could choose exactly which aspect of the language and on which level of difficulty she would like to train. To better integrate the tools to the instruction, the vocabulary was designed so that it may be restricted to particular textbooks. Finally, the programs were made freely accessible via Internet.

Due to its complex morphology, Sámi languages demand a lot of practising before the student reaches a level of fluency required for everyday conversation. Since Sámi is a minority language, learners often do not have enough opportunities to practise the language in a natural setting. Our programs give a practical supplement to the instruction given at school or university. In addition, the dialogue program consists of everyday topics, with underlying pedagogical goals such as practicing verb inflection, choice of correct case form or vocabulary learning.

The student may choose between two main North Sámi dialects. Especially when training morphology, it is important that the forms that are presented for the user, are the same that the ones used in the language society or taught during instruction. Still, the program accepts any correct orthographic word form provided by the student.

North Sámi is used in three countries, and therefore the programs have several metalanguages (Norwegian, Finnish, North Sámi, English). We are also considering extending the programs to other Sámi languages.

3 Pedagogical lexicon

3.1 The structure of the lexicon

All the OAHPA! programs share a set of common resources: a pedagogical lexicon and a morphological generator that is used for generating the different word forms that appear in the programs. The dialectal variation is taken into account in the lexicon as well as in the morphology. In addition, the morphological properties of words are used when making a detailed feedback on morphological errors.

The pedagogical lexicon forms a collection of words that are considered relevant for the learners of North Sámi in schools and universities. The words occur in different forms in the tasks. The pedagogical lexicon contains additional information about the lemmata, such as Norwegian and Finnish translation, semantic class, dialect and information about the inflection. The words in the pedagogical lexicon were collected from the key textbooks for North Sámi and the source information is included in the lexicon entry. In addition, homonymy in both base form and inflection is dealt with using ids for lexicon entries instead of lemmata. The lexicon consists of 1538 nouns, 500 verbs and 194 adjectives, in addition to a small lexicon for closed parts of speech. Figure 1 shows an example of an entry in the noun lexicon.

The word forms that are used in the program are pre-generated with a transducer that contains of the full North Sámi vocabulary, the inflectional and derivational morphology, and the non-segmental morphological processes (consonant gradation, diphthong simplification, etc.). Similar transducer is used in live analysis of user input in the programs Vasta and Sahka, which are described in section 4.

The contents of the pedagogical lexicon as well as full paradigms for each lexicon entry are stored in the Mysql database. The database allows effective processing of queries and multiple simultaneous users. In addition, generating the word forms

```
<entry id="monni">
  <lemma>monni</lemma>
  <pos class="N"/>
  <translations>
    <tr xml:lang="nob">egg</tr>
    <tr xml:lang="fin">muna</tr>
  </translations>
  <semantics>
    <sem class="FOOD-GROCERY"/>
  </semantics>
  <stem class="bisyllabic" diphthong="no"
     gradation="yes" soggi="i" rime="0"/>
  <dialect class="NOT-KJ"/>
  <sources>
    <book name="d1"/>
    <book name="sara"/>
    <book name="algu"/>
  </sources>
</entry>
```

Figure 1: An entry in the pedagogical lexicon.

and storing them to the database provides better control over the inflected word forms and e.g. different dialectal forms. The handling of dialectal variation is described in the next section.

3.2 Handling the dialectical variation

When generating sentences or providing the correct answers for the user, we wanted to control the selection of word forms to allow only normative forms in the correct dialect. On the other hand, the live analyser used for the analysis of the user input should be tolerant and accept all correct variants of the same grammatical word. Therefore we compiled different analysers/generators for different purposes: one normative but variation-tolerant transducer for analysing the input, and two strict ones for different dialects for sentence generation.

The variation between the main dialects Kárášjohka and Guovdageaidnu was in the source code (lexc) marked in one of the following ways:

(a) NOT-KJ (not generated for KJ-dialect)

(b) NOT-GG (not generated for GG-dialect)

We also marked entries in the pedagogical lexicon-files as in Figure 1. This system can easily be expanded with more dialects. Figure 2 contains an example of how the dialectal information is handled in the morphological analyser.

3.3 Feedback on morphological errors

The inflectional information of words contained in pedagogical lexicon is used for generating feedback to the student. If the user does not inflect the lemma correctly, she can ask for hints about the inflection, and try once more, instead of getting the correct answer straight away.

```
+A+Comp:i%>X4b BUStem ; !  NOT-KJ
+A+Comp:á%>X4b BUStem ; !  NOT-GG
```

Figure 2: Handling of dialectal variation in the morphological analyser.

The feedback messages are determined by the combination of morphological features in the lexicon and the inflection task at hand. Consider a part of the feedback specification in the Figure 3. It specifies the morphological rule that there is a vowel change in illative singular for bisyllabic nouns that end with the vowel *i*. The corresponding feedback message instructs the user to remember the vowel change.

```
<stem class="bisyllabic" soggi="i">
  <msg case="Ill" number="Sg">i_á</msg>
  <note>láibi > láibái </note>
</stem>

<message id="i_á">Vowel change i > á.
</message>
```

Figure 3: The features in the lexicon are used to determine the correct feedback message, in this case the message is ”Vowel change i > á”.

The feedback may consist of several parts so that the user also receives information about e.g. stem class. All the feedback messages that match the feature definition in the given task, are collected and given to the user in a specified order.

4 CG-parser in live analysis programs Vasta and Sahka

4.1 Syntactic analyses of the student's answer

We have chosen not to use multiple-choice, but rather let the student formulate her own answer. To a certain question one may give many kinds of acceptable answers. In Sámi one may change word order, and also add many kinds of particles.

We use vislcg3 for analysing the student's answer. The reason for choosing CG as parser platform was that only CG is robust enough for handling unconstrained input, and at the same time accurate enough to identify errors. The program contains manually written, context dependent rules, mainly used for selecting the correct analysis in case of homonymy. Each rule adds, removes, selects or replaces a tag or a set of grammatical tags in a given sentential context. Context conditions may be linked to any tag or tag set of any word anywhere in the sentence, either locally (in a fixed subdomain of the context) or globally (in the whole context). Context conditions in the same rule may be linked, i.e. conditioned upon each other, negated or blocked by interfering words or tags. Vislcg3 is documented at (visl, 2008). Grammars for Danish and Norwegian based on CG achieve very good F-scores (Bick, 2003).

The question and the answer are merged, and given to the analyser as one text string. We use a ruleset file which disambiguates the student's input only to a certain extent, because there will probably be grammatical and orthographic errors. The last part of the file consists of rules for giving feedback to the student's grammatical errors, and rules for navigating to the correct next question of in the dialogue, due to the student's answer. How to generate feedback or navigation instructions is explained in section 4.2 and 4.6.

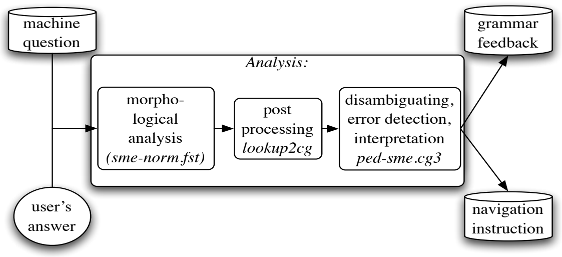
machine question
Analysis:
morpho-logical analysis (sme-norm.fst)
post processing lookup2cg
disambiguating, error detection, interpretation ped-sme.cg3
grammar feedback
navigation instruction
user's answer

Figure 4: An overview of the analysis process.

The question mark is exchanged for a special symbol ("qst" QDL), cf. figure 5. Instead of a sentence delimiter, we want to be able to refer to the question and the answer separately in the rules.

4.2 Tutorial feedback

Tutorial feedback is feedback about grammar errors (CG prefix *&grm*), and in Figure 6 we see a rule for assigning a tag if the student has not used accusative, when the question requires it. If the interrogative pronoun is in accusative, we expect an accusative in the answer. The rule assigns a *&grm-missing-Acc* tag to the interrogative pronoun if there is no accusative or negation verb in the answer.

```
"<Maid>"
    "maid" Adv
    "maid" Interj
    "mii" Pron Interr Pl Acc
    "mii" Pron Interr Sg Acc
    "mii" Pron Rel Pl Gen
    "mii" Pron Interr Pl Gen
    "mii" Pron Rel Sg Acc
    "mii" Pron Rel Pl Acc
"<don>"
    "dot" Pron Dem Sg Gen
    "don" Pron Pers Sg2 Nom
    "dot" Pron Dem Sg Acc
"<lohket>"
    "lohkat" V TV Ind Prs Pl3
    "lohkat" V TV Imprt Prs Pl2
    "lohkat" V TV Ind Prt Sg2
"<ikte>"
    "iktit" V TV Ind Prt Pl3
    "iktit" V TV Ind Prs Du1
    "ikte" Adv
"<^qst>"
    "^qst" QDL
"<Ikte>"
    "iktit" V TV Ind Prt Pl3
    "iktit" V TV Ind Prs Du1
    "ikte" Adv
"<mun>"
    "mun" Pron Pers Sg1 Nom
"<lohken>"
    "lohkat" V TV Ind Prt Sg1
"<boares>"
    "boaris" A Attr
"<girji>"
    "girji" N Sg Nom
"<.>"
    "." CLB
```

Figure 5: Between analysis and disambiguation.

```
LIST TARGETQUESTION-ACC = ("mii" Acc) ("gii" Acc)
("galle" Acc) ("gallis" N Acc) ;

MAP (&grm-missing-Acc) TARGET TARGETQUESTION-ACC IF
(*1 QDL BARRIER WORK-V LINK NOT *1 Acc OR Neg
BARRIER S-BOUNDARY);
```

Figure 6: Rule assigning missing Acc -tag.

Figure 7 shows how the vislcg3 file has disambiguated and added the error tag to the input which is the analysis from Figure 5. The tag generates feedback to the student. The object is in Nom instead of Acc, and the grammar adds the error tag.

The most difficult problem for the grammatical analysis are the student's misspellings. A misspelling may be left unrecognized in the analysis or it can produce another word form for the same lemma, or from some other lemma.

When the word form is not recognized during the analysis, the feedback message to the student points to the unrecognized word form asking the student to check the spelling. To the extent that misspellings are the most common type of errors, the current feedback does not provide enough in-

```
"<Maid>"
   "mii" Pron Interr Pl Acc &grm-missing-Acc
   "mii" Pron Interr Sg Acc &grm-missing-Acc
"<don>"
   "don" Pron Pers Sg2 Nom
"<lohket>"
   "lohkat" V TV Ind Prt Sg2
"<ikte>"
   "ikte" Adv
"<^qst>"
   "^qst" QDL
"<Ikte>"
   "ikte" Adv
"<mun>"
   "mun" Pron Pers Sg1 Nom
"<lohken>"
   "lohkat" V TV Ind Prt Sg1
"<boares>"
   "boaris" A Attr
"<girji>"
   "girji" N Sg Nom
"<.>"
   "." CLB

<message id="grm-missing-Acc">The answer should
   contain an accusative.</message>
```

Figure 7: QA with missing Acc -tag added because the object *girji* is in Nom (What did you read yesterday? Yesterday I read an old book-SgNom).

structions for the student to improve the spelling. However, in order to give better feedback to certain misspellings, we have added e.g. place names with small initial letter to the fst, together with an error tag, so that the student gets a precise feedback. We will implement more that kind of rules and consider usage of a spell checker to help the student to find the correct word form.

For misspellings that produce another word form of the same lemma, we have written rules that are based on the grammatical context. The real problem emerges when the spelling error gives rise to an unintended lemma. Then the challenge is to give a feedback according to what the student thinks she has written. In this case, feedback has to be tailored using the knowledge about the student's interlanguage. We have created sets for typical unintended lemmata. Combined with contextual rules we can then give the user a good feedback due to the misspelling instead of the unintended lemma.

E.g. if the student uses the Sg2 form of the main verb after the negative verb, instead of the correct ConNeg form, then the erroneous form can be a ConNeg form of a derivated verb, and the normal feedback will be: ”You should answer with the same verb as in the question.” The student will not understand this, because she thinks that the word form in the answer is an instance of the same verb. The solution was to generate all these forms of the verbs in the questions, make a set of them, and make a rule for in the right context, give the feedback: ”The negative form is not correct.”

4.3 The open QA drill – Vasta

In between the ”natural” dialogues, mimicking real life dialogues, and the pure grammar training session, inquiring paradigm forms, we have made a question-answer drill. The drill has two question types: Yes/no questions and wh-questions.

There are two motives for making this program type. First, our tailored dialogues run the risk of getting quickly consumed. With a QA drill we may generate an indefinite number of questions. Second, the students need to automate the question-answer routine – answer with the correct verb, inflect the finite verb correctly and choose the correct case form.

The questions are generated, and then the question and answer are analysed together, and the student gets feedback, as described in 4.1. The question matrices are marked with level, so there is a level option. Only one question is presented at a time. The student can answer what she wants, but she has to use a full sentence (containing a finite verb), and use the same verb as in the question. There are 111 matrix questions.

4.4 Sentence generator

One of the main goals of the programs in OAHPA! is to practice language in natural settings with variation in the tasks. In order to provide variation in programs that involve sentential context we implemented a sentence generator. The sentence generator is used in the morphology in sentential context program (Morfa-C), and for generating questions to the QA drill (Vasta). Figure 8 contains an example of sentence matrix that is used in the sentence generator.

The question matrix contains two types of elements: constants and grammatical units. The constants such as *go* and *ikte* in the Figure 8 are present in each generated sentence as such, whereas grammatical units allow more variation. Both the inflection and the content of the grammatical units may vary from question to question, and from program to program. E.g. in the question in Figure 8 the MAINV is fixed to past tense, but the person and number inflection may vary freely. In addition, certain elements such as the sentence

```
<q level="2" id="go_ikte">
  <qtype>PRT</qtype>
  <question>
    <text>MAINV go SUBJ ikte</text>
    <element id="MAINV">
  <grammar tag="V+Ind+Prt+Person-Number"/>
  <sem class="ACTIVITY"/></element>
    <element id="SUBJ">
  <sem class="HUMAN"/>
  <grammar pos="N"/></element></question>
</q>
```

Figure 8: Example showing question generation (MAINV question-particle SUBJ yesterday).

subject (SUBJ) have default inflection in nominative, but the default inflection may be overridden. The selection of words for the sentence is constrained by semantic sets. Semantic sets are also used as an option in the word quiz (Leksa).

The sentence generator handles agreement e.g. between subject and the main verb. The agreement may be explicitly marked between any two elements, which indicates that the two elements share the same number and person inflection.

In addition to generating questions, the sentence generator is used for generating answer templates. In this case, the sentence generator takes into account the agreement inside a sentence, but also the content and agreement between the question and the answer. For example, the person and number inflection in the answer is restricted by the question. We chose not to accept an inclusive interpretation of the pronouns in Pl1 and Du1, because we wanted the student to exercise also 2. person verb inflection. Table 1 shows how the question Person-Number (QPN) Sg1 requires answer Person-Number (APN) Sg2, and so on. Pl1 as an answer to Pl1 is thus not accepted by the system.

Table 1: Provided question-answer agreement.

QPN	APN	QPN	APN	QPN	APN
Sg1	Sg2	Du1	Du2	Pl1	Pl2
Sg2	Sg1	Du2	Du1	Pl2	Pl1
Sg3	Sg3	Du3	Du3	Pl3	Pl3

4.5 The dialogue program – Sahka

The idea behind the dialogue program is that the student may exercise North Sámi in a natural setting, and at the same time receive feedback about errors. Each dialogue is based on a scenario, such as meeting a person for the first time or going to a grocery store. In addition, each scenario has a set of underlying pedagogical goals. E.g. in the Grocery-dialogue the student is telling what kind of food she wants to buy and the underlying pedagogical goal is to exercise inflecting objects in accusative.

Each dialogue consists of branches to different topics. The program asks questions, comments on the student's answers and starts a new topic according to the answer. The dialogue forms a continuum and contains only accepted answers. The feedback concerning grammatical errors is given on a separate window and the user is allowed to correct the answer until it is accepted.

A topic starts with an opening utterance which is either a question or a comment followed by a question. Thus, the user expected to provide answers to the questions throughout the dialogue. The dialogue proceeds to an appropriate utterance inside the current topic. In the end of the topic, there is always a closing comment after which the dialogue proceeds to next topic. Both the next utterance and the next topic may be selected based on the information in the user's answer. For example, if the question is about having a car, a positive answer will navigate to a branch with a follow-up questions. In the next section, we describe the navigation inside the dialogue in more detail.

The dialogue system itself is quite simple. Only the program can make initiatives, and all the utterances from the program are ready-made, addressing topics that the program is able to handle. In other words, the sentence generation mechanism used in Vasta is not utilised in the dialogue program. Developing the program to the direction of free dialogue, where also the student is able to take initiatives, requires among other things an analyser which maps semantic roles to the student's input and a semantically enriched lexicon.

4.6 Navigating in the Sahka dialogue

Navigating inside the dialogue is implemented in CG-rules. The user input is tagged during analysis with information on whether the answer is interpreted as affirmative or negative. In addition, a special tag indicates whether the sentence contains some information that should be stored for the following questions or utterances. The program is thus able to store simple information such as the student's name, place where she lives and for example the type of her car and use this information in tailored utterances.

Every utterance contains one or more links to other utterances. The link is selected according to the tag assigned to the question-answer pair, e.g. *&dia-neg* for a negative answer, *&dia-pos* for a positive answer, or *&dia-target* for a certain word, e.g. target="hivsset", like in Figure 9. In Figure 10 we see how the *&dia-target* tag is mapped to the noun in illative. The question is "In which room do we put the TV?" One of the alternatives for the navigation is due to the target tag being assigned to the lemma *hivsset* ("WC"). The answer will be "That is not a good idea. Make a new try."

```
<utt type="question" name="gosa_bidjat_TV">
  <text>Guđe latnjii moai bidje mu TV?</text>
  <alt target="hivsset" link="gosa_bidjat_TV">
  <text>Dat gal ii heive! Geahččal ođđasit.</text>
        </alt>
  <alt target="default" link="gosa_bidjat_beavddi">
<text>Moai gudde dan ovttas dohko.</text></alt></utt>
```

Figure 9: Rule for navigating according to answer.

Figure 10 shows a general rule, not connected to any particular question, for adding a target-tag to the NP-head in illative after a question with the interrogate *guhte* + a noun in illative (= "to which").

```
MAP (&dia-target) TARGET NP-HEAD + Ill IF
(*-1 QDL BARRIER S-BOUNDARY LINK **-1 (N Ill)
LINK -1 ("guhte"))(NOT 0 NOTHING) ;
```

Figure 10: Case tag adding triggered by question.

Every question has its own unique id, which is used in navigating between questions. In addition, the CG-rules may be tailored for specific questions. An answer from the student about her age will induce a tag (Figure 11), which functions as a link when moving to the next dialogue branch. Figure 12 gives an example of how to navigate to the next question or branch, with help of the tag. The question introducing the choice is "How old are you?"

5 Evaluation

At the time of writing, the programs have been in public use for approximately two months. All user input the word quiz Leksa, the numeral quiz Numra, the bare morphological task Morfa-S and

```
# Adding age-tags
MAP (&dia-adult) TARGET Num (*-1 QDL LINK 0
(Man_boaris_don_leat))(0 ("([2-9][0-9])"r))  ;
MAP (&dia-young) TARGET Num (*-1 QDL LINK 0
(Man_boaris_don_leat))(0 ("([1][0-9])"r))  ;
MAP (&dia-child) TARGET Num (*-1 QDL LINK 0
(Man_boaris_don_leat))(0 ("([1-9])"r))  ;
```

Figure 11: Rules for giving age-tag to the input.

```
<utt type="question" name="Man_boaris_don_leat">
  <text>Man boaris don leat?</text>
  <alt target="young" link="at_school_young"/>
  <alt target="child" link="begin_school_child"/>
  <alt target="adult" link="job_adult"/>
  <alt target="default" link="job_adult"/>
</utt>
```

Figure 12: Navigating to the next question or branch, with help of a tag.

the contextual morphology task Morfa-C has been logged from the very beginning. Unfortunately the programs Vasta and Sahka, have been logged for a couple of days only. The log contains 32475 queries (679 queries/day for the 4 programs logged the whole period), of these, approximately 600, or under 2%, were nonsense answers.

Table 2: Answers to the programs (Vasta and Sahka were logged at the end of the period only).

Program	Correct	Wrong	Total	%
Morfa-S	6920	6323	13243	52.3
Leksa	5659	4248	9907	57.1
Numra	3086	2512	5598	55.1
Morfa-C	1349	1613	2962	45.5
Sahka	322	322	644	50.0
Vasta	19	102	121	15.7
Total	17355	15120	32475	53,44

As can be seen from Table 2, slightly more than half of the queries resulted in correct answers. When confronted with an error feedback, the user is offered grammatical help, and thereafter she has the possibility to give a new answer to the same query. An investigation of 1500 queries to Morfa-C showed that 444, or 30%, were such repeated answers. Even though we have no log info of the use of the morphological feedback (section 3.3), our impression from classroom experience is that the users are actively using the feedback system. This indicates that what we are witnessing is a truly interactive process, where users err in half of the queries, and then follow up with a new try, possibly after having read the morphological advice from the program.

The error log for Sahka shows that one fourth of the errors are due to orthographical errors (Table 3). Most of the ”no finite verb” errors are elliptical answers, and these are not accepted, for pedagogical reasons. The remaining cases are errors where the misspelled verb is an existing word. Also for the other grammatical errors verb errors are dominating. The main goal of the program was to train verb forms in a dialogue, and the error log shows that the program is able to capture such errors.

The logs may not only be used for evaluating the programs, but also for monitoring the learning process as such. To take just one example, the Morfa logs give the error rate for each and every morphosyntactic property and stem type, thereby giving valuable information as to which parts of the verbal paradigm are the most problematic ones.

Table 3: Error types for Sahka, ordered after type.

Error type	#	Error type	#
no finite verb	85	wr. case for V-arg	22
orth. error	83	wr. case after Num	10
wrong S-V agr	46	wrong tense	9
no infinite V	30	no postposition	6
wrong V choice	24	wrong word	7

6 Conclusion

By using a sloppy version of the syntactical analyser for North Sámi, combined with a set of error-detection rules, we have been able to build a flexible CALL resource. The programs are modular, and the modules may be improved by adding more materials – words, tasks, dialogues, levels, words from textbooks. The CG parser framework was originally chosen as parser framework for Sámi due to its extraordinary results for free-text parsing. The present project has shown that CG is well fit for making pedagogical dialogue systems as well.

The program suite is something quite new among pedagogical programs for Sámi, and indeed its dialogue and open QA-programs are quite rare within the field of parser-based CALL. The QA and the dialogue program are tolerant towards variation in student answer (not only string matching), and the random generation of tasks more or less in all of the programs allows the student to use them over and over again.

Acknowledgments

Thanks to the faculty of Humanities at the University of Tromsø, and the Sámi Parliament in Norway, for funding the project.

References

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI publications in Computational Linguistics. USA.

Eckhard Bick. 2003. PaNoLa: Integrating Constraint Grammar and CALL applications for Nordic languages. Holmboe, Henrik (ed.): *Nordic Language Technology, Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*. 183–190, København: Museum Tusculanums Forlag.

Eckhard Bick. 2005. Live use of Corpus data and Corpus annotation tools in CALL: Some new developments in VISL. Holmboe, Henrik (ed.): *Nordic Language Technology, Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*, 171–185. København: Museum Tusculanums Forlag.

Johann Gampfer and Judith Knapp. 2001. A review of intelligent CALL systems. *Computer Assisted Language Learning* 15(4):329–342.

Trude Heift. 2001. Intelligent Language Tutoring Systems for Grammar Practice. *Zeitschrift fur Interkulturellen Fremdsprachenunterricht [Online]* 6(2).

Trude Heift and Devlan Nicholson. 2001. Web Delivery of Adaptive and Interactive Language Tutoring. *International Journal of Artificial Intelligence in Education* 12(4):310–325.

Trude Heift and Mathias Schulze. 2007. *Errors and intelligence in computer-assisted language learning: parsers and pedagogues*. Routledge studies in computer-assisted language learning 2. New York : Routledge.

Fred Karlsson and Atro Voutilainen and Juha Heikkilä and Arto Anttila. 1995. *Constraint grammar: a language-independent system for parsing unrestricted text*. Mouton de Gruyter.

Trond Trosterud. 2007. *Language technology for endangered languages: Sámi as a case study*. http://giellatekno.uit.no/background/rvik.pdf University of Tromsø, Norway.

VISL-group. 2008. *Constraint Grammar*. http://beta.visl.sdu.dk/constraint_grammar.html University of Southern Denmark.